Strutture dei Sistemi Operativi

- Componenti di sistema
- Servizi del sistema operativo
- Chiamate di sistema
- Programmi di sistema
- Struttura del sistema
- Macchine virtuali
- Progetto e implementazione di sistemi
- Generazione del sistema

Componenti di sistema

I sistemi operativi mettono a disposizione l'ambiente in cui i programmi vengono eseguiti.

- Gestione dei processi
- Gestione della memoria centrale
- Gestione del file system
- Gestione del sistema di I/O
- Gestione della memoria secondaria
- Networking
- Tecniche di protezione
- Interprete dei comandi

Gestione dei processi

- **Un *processo* è un programma in esecuzione**. Un processo necessita di alcune risorse per assolvere il proprio compito: tempo di CPU, memoria, file e dispositivi di I/O.
- Il processo è l'unità di lavoro di un sistema.
- Il SO è responsabile delle seguenti attività relative alla gestione dei processi:
  - Creazione e cancellazione di processi (utente e di sistema).
  - Sospensione e riattivazione di processi.
  - Fornire meccanismi per:
    - ✓ sincronizzazione di processi;
    - ✓ comunicazione fra processi;
    - ✓ gestione dei *deadlock*.

Gestione della memoria centrale

- La memoria è una grande matrice di parole o byte, ciascuna con un proprio indirizzo. È un deposito di dati rapidamente accessibili condiviso dalla CPU e dai dispositivi di I/O.
- La memoria centrale è un dispositivo di memorizzazione volatile. Perde il suo contenuto per guasti del sistema.
- Il sistema operativo è responsabile delle seguenti attività connesse alla gestione della memoria centrale:
  - Tener traccia di quali parti della memoria sono attualmente usate e da chi.
  - Decidere quali processi caricare in memoria quando vi è spazio disponibile.
  - Allocare e deallocare lo spazio di memoria in base alle necessità.

Gestione del file system

- Per rendere conveniente l'utilizzo del computer, il SO fornisce una visione logica uniforme del processo di memorizzazione: astrae dalle caratteristiche fisiche dei dispositivi per definire un'unità di memorizzazione logica, il *file*.
- Un file è una collezione di informazioni correlate definite dal loro creatore. Comunemente i file rappresentano programmi (sia sorgente che oggetto) e dati.
- Il sistema operativo è responsabile delle seguenti attività connesse alla gestione di file:
  - Creazione e cancellazione di file.
  - Creazione e cancellazione di directory.
  - Supporto alle funzioni elementari per la manipolazione di file e directory.
  - Associazione dei file ai dispositivi di memoria secondaria.
  - Backup di file su dispositivi di memorizzazione stabili (non volatili).

Gestione del sistema di I/O

- Uno scopo fondamentale del SO: nascondere all'utente le caratteristiche di specifici dispositivi hardware.
- Il sottosistema di I/O di UNIX nasconde le caratteristiche dell'hardware al SO stesso e consiste di:
  - Un componente di gestione della memoria comprendente il buffering, il caching e lo spooling.
  - Un'interfaccia generale per i driver dei dispositivi.
  - Driver per gli specifici dispositivi hardware.
- Soltanto il driver di dispositivo conosce le caratteristiche dello specifico dispositivo cui è assegnato.

Gestione della memoria secondaria

- Poiché la memoria centrale è volatile e troppo piccola per contenere tutti i dati e i programmi permanentemente, il sistema di elaborazione deve consentire un'archiviazione secondaria per salvare i contenuti della memoria centrale.
- La maggior parte dei moderni sistemi di elaborazione impiega i dischi come principale mezzo di memorizzazione on–line, sia per i programmi che per i dati.
- Il SO è responsabile delle seguenti attività relative alla gestione dei dischi:
  - Gestione dello spazio libero.
  - Allocazione dello spazio.
  - Scheduling del disco.

Networking (Sistemi distribuiti)

- Un sistema ***distribuito*** è un insieme di processori che non condividono né la memoria né il clock. Ciascun processore ha la sua propria memoria locale.
- I processori nel sistema sono connessi attraverso una rete di comunicazione.
- La comunicazione avviene secondo un dato *protocollo.*
- Un sistema distribuito fornisce agli utenti l'accesso a varie risorse di sistema.
- L'accesso a risorse condivise consente di:
  - Accelerare l'elaborazione.
  - Aumentare la disponibilità di dati.
  - Migliorare l'affidabilità.

Tecniche di protezione

- ***Protezione*** — è il meccanismo usato per controllare l'accesso da parte di processi o utenti a risorse del sistema di calcolo (di sistema e di altri utenti).
- Il meccanismo di protezione deve:
  - Distinguere fra uso autorizzato e non autorizzato.
  - Specificare i controlli da imporre.
  - Fornire una modalità di imposizione.

Interprete dei comandi

- Molti comandi sono impartiti al sistema operativo per mezzo di *istruzioni di controllo* che hanno a che fare con:
  - creazione e gestione dei processi;
  - gestione di I/O;
  - gestione della memoria secondaria;
  - gestione della memoria centrale;
  - accesso al file system;
  - protezione;
  - comunicazione su rete.
- Esistono interpreti “amichevoli” con finestre, o interpreti più potenti (ma più complessi) basati su interfaccia a carattere.

Interprete dei comandi

- È il programma che legge ed interpreta le istruzioni di controllo; viene chiamato in vari modi:
  - interprete della linea di comando
  - ***shell*** (in UNIX)

  La sua funzione è quella di riconoscere ed eseguire la successiva istruzione di comando.

Servizi del sistema operativo

- **Esecuzione di programmi** — capacità di caricare un programma in memoria e mandarlo in esecuzione.
- **Operazioni di I/O** — poiché i programmi non possono eseguire le operazioni di ingresso/uscita direttamente, il SO deve fornire i mezzi per effettuare l'I/O.
- **Manipolazione del file system** — capacità dei programmi di leggere, scrivere e cancellare file.
- **Comunicazioni** — scambio di informazioni fra processi in esecuzione sullo stesso elaboratore o su sistemi diversi, connessi per mezzo di una rete. Implementate per mezzo di ***memoria condivisa*** o ***scambio di messaggi***.
- **Rilevamento di errori** — assicura una corretta elaborazione rilevando errori nella CPU e nella memoria, in dispositivi I/O o in programmi utente.

Altre funzioni del SO

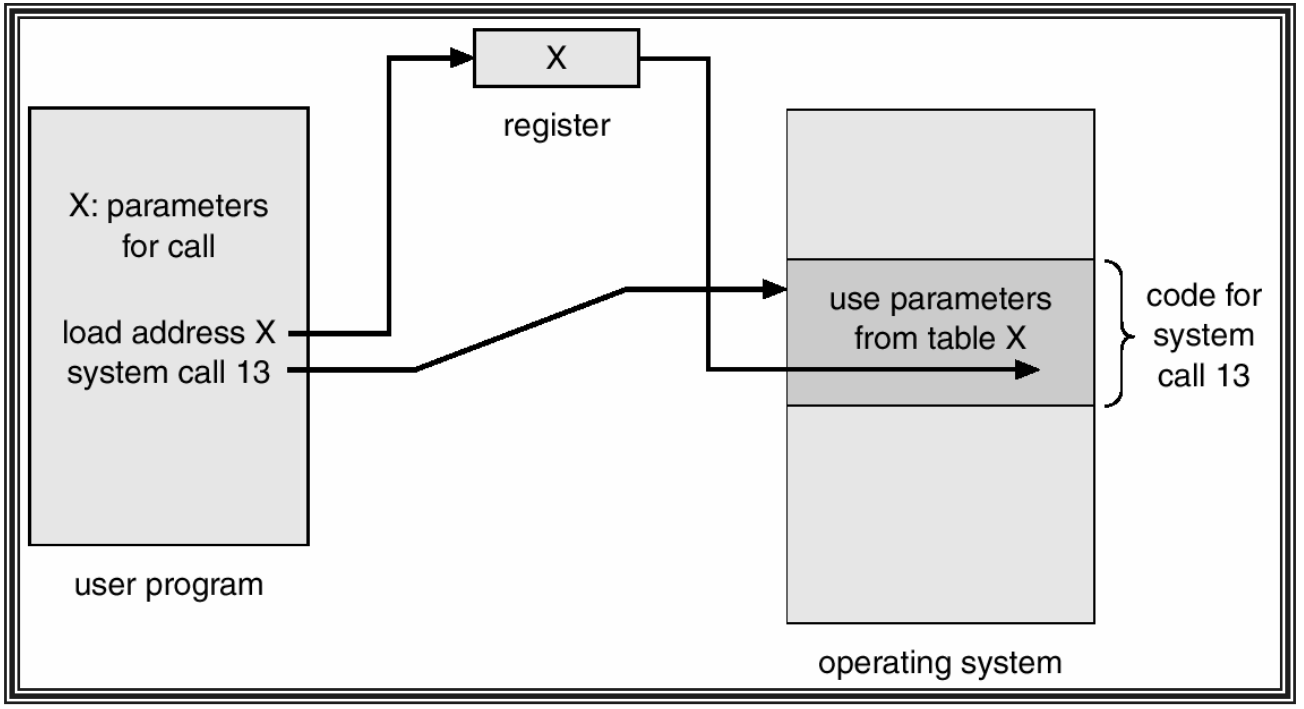
Esistono funzioni addizionali atte ad assicurare l'efficienza delle operazioni di sistema piuttosto che orientate all'utente.

- **Allocazione di risorse** — a più utenti o a job multipli in esecuzione contemporanea.
- **Contabilizzazione dell'uso delle risorse** — tener traccia di quali utenti usano quali e quante risorse del sistema. Queste informazioni sono memorizzate per addebitare il costo delle risorse, o per accumulare statistiche di uso.
- **Protezione** — assicurare che tutti gli accessi alle risorse di sistema siano controllati. La sicurezza di un sistema comincia con l'obbligo di identificazione tramite *password* e si estende alla difesa dei dispositivi di I/O esterni (modem, adattori di rete, etc.) da accessi illegali.

Chiamate di sistema

- Le chiamate al sistema forniscono l'interfaccia fra un programma in esecuzione e il sistema operativo.
  - Sono generalmente disponibili come istruzioni in linguaggio Assembler.
  - Alcuni linguaggi definiti al fine di sostituire il linguaggio Assembler per la programmazione dei SO permettono di effettuare le chiamate di sistema direttamente (ad es., C, C++).
- Tre metodi generali sono impiegati per passare i parametri tra un programma in esecuzione e il sistema operativo.
  - Impiego dei *registri* (passaggio di parametri tramite registri).
  - Memorizzazione dei parametri in una tabella in memoria, e passaggio dell'indirizzo della tabella come parametro in un registro.
  - *Push* dei parametri nello stack da parte del programma. Il SO recupera i parametri con un *pop*.

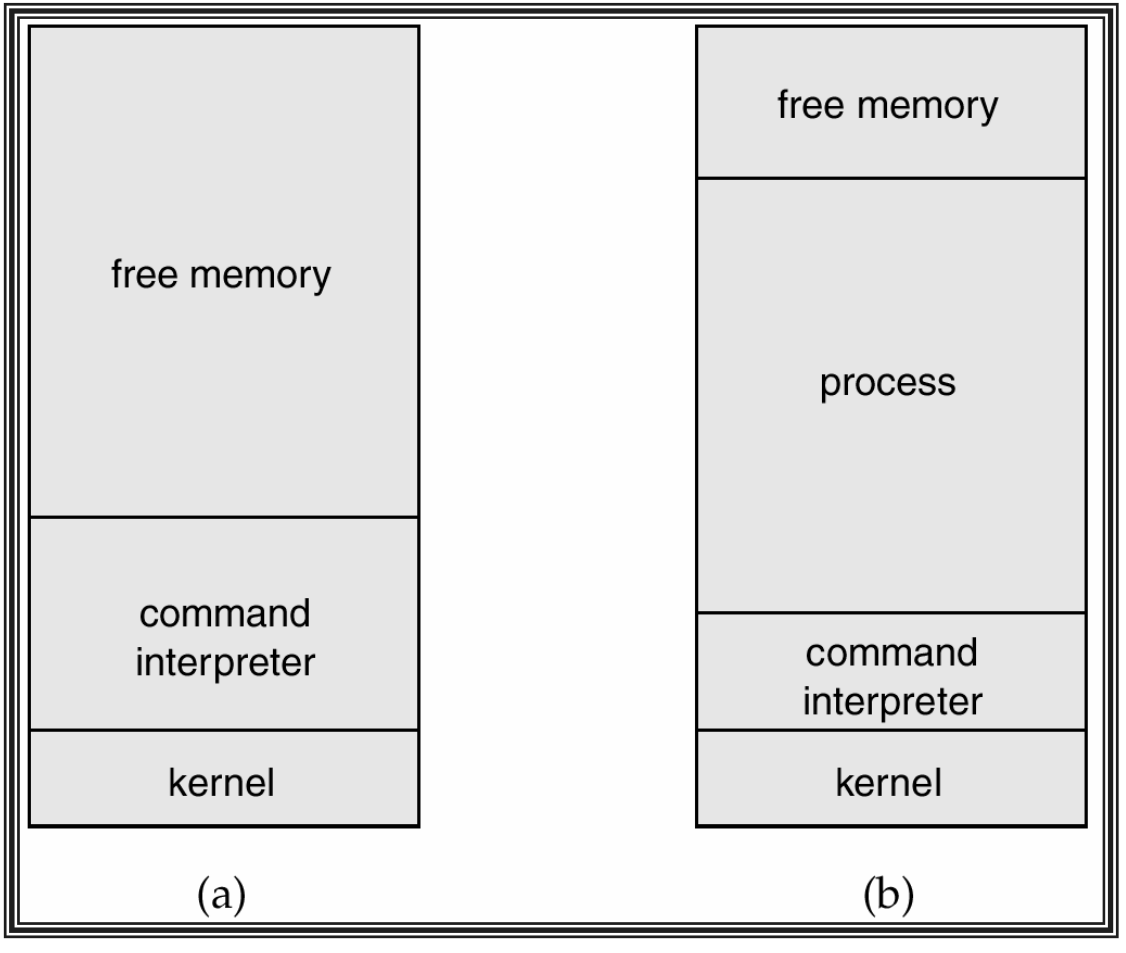
Passaggio di parametri tramite tabella



Tipi di chiamate di sistema

- **Controllo di processo**: *end, abort, load, execute, create/terminate process, get/set process attributes, wait/signal event, allocate/free mem.*
- **Manipolazione dei file**: *create/delete file, open, close, read, write, reposition, get/set file attributes*.
- **Gestione dei dispositivi**: *request/release device, read, write, reposition, get/set device attributes, attach/detach devices.*
- **Gestione delle informazioni**: *get/set time/date, get/set system data, get/set file/device attributes*.
- **Comunicazione**: *create/delete communication connection, send/receive messages, transfer status information.*

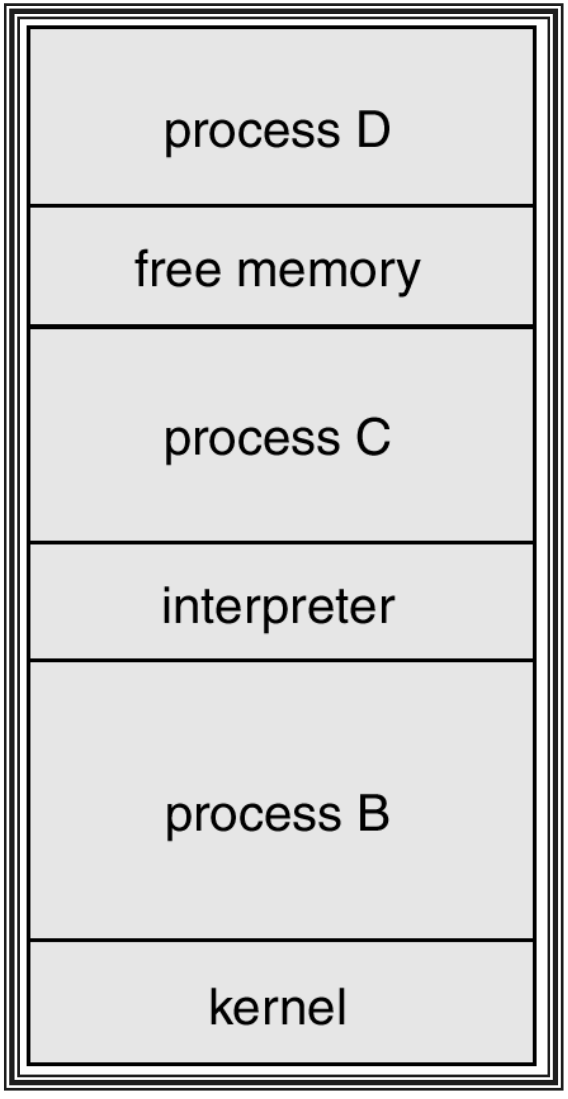
Controllo dei processi in MS–DOS



Allo Start-up del sistema

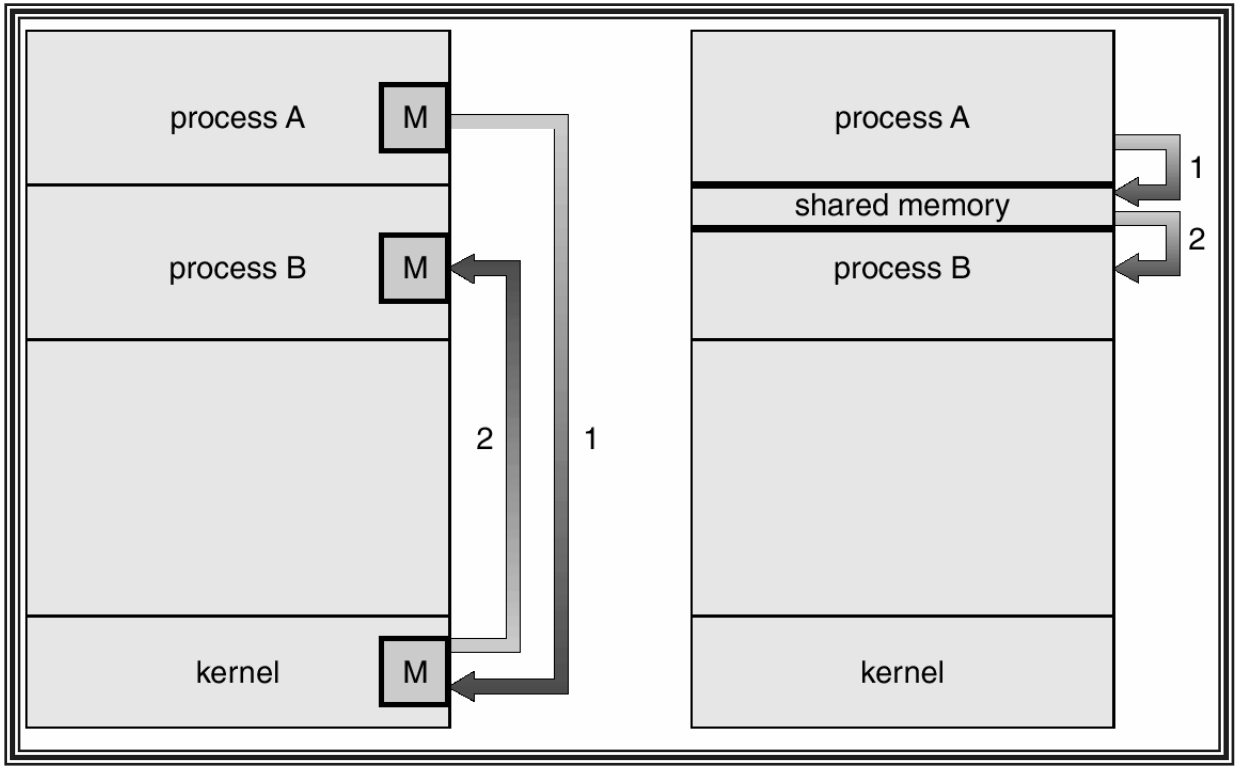
Durante l'esecuzione di un programma

UNIX con più programmi in esecuzione



Modelli di comunicazione

- La comunicazione può aver luogo sia attraverso scambio di messaggi che con l'utilizzo di memoria condivisa.



Scambio di messaggi Memoria condivisa

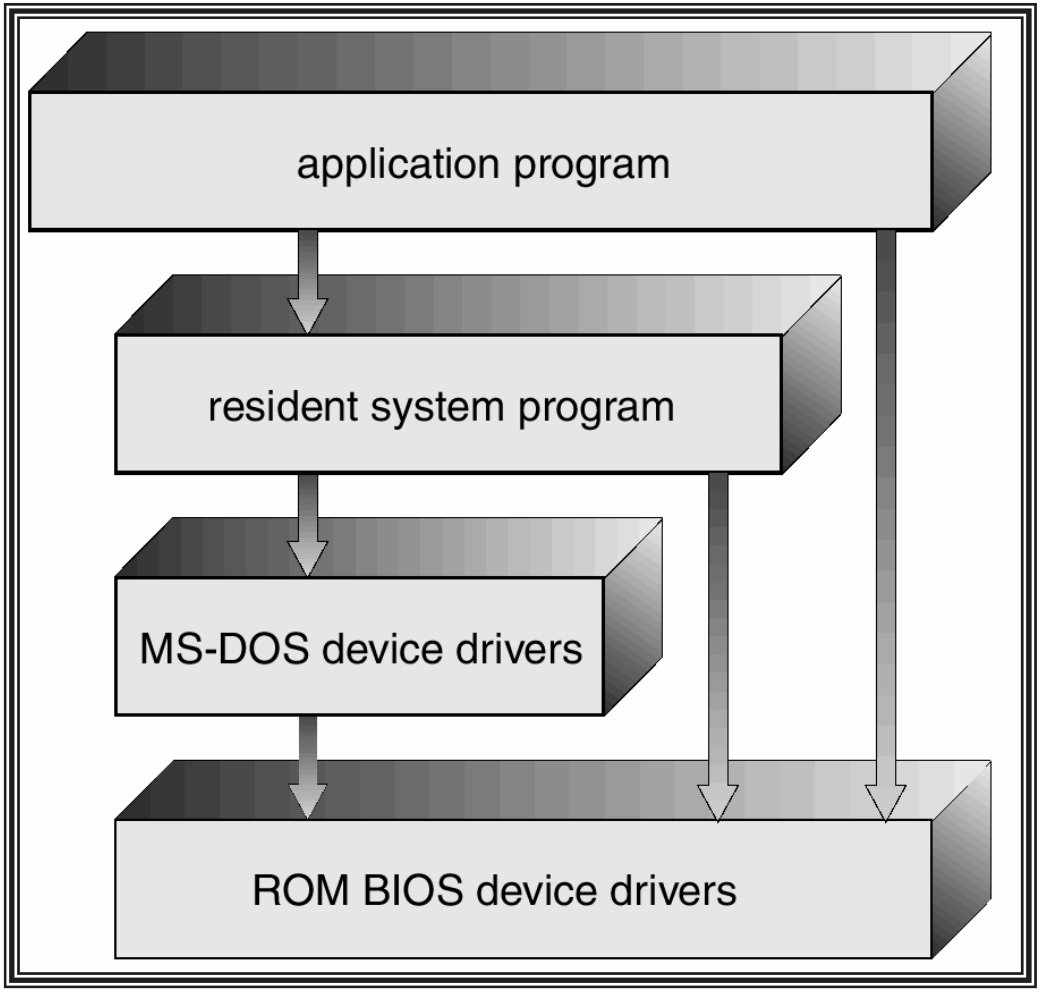
Programmi di sistema

- I programmi di sistema forniscono un ambiente conveniente per lo sviluppo e l'esecuzione di programmi. I programmi possono essere suddivisi in:
  - Manipolazione di file
  - Informazioni sullo stato
  - Modifica di file
  - Supporto a linguaggi di programmazione
  - Caricamento ed esecuzione di programmi
  - Comunicazioni
  - Programmi applicativi
- L'aspetto del SO per la maggioranza degli utenti è definito dai programmi di sistema, non dalle chiamate di sistema vere e proprie.

Struttura del sistema MS–DOS

- **MS–DOS** — scritto per fornire il maggior numero di funzionalità utilizzando la minor quantità di spazio possibile:
  - non è suddiviso in moduli;
  - sebbene MS–DOS abbia una qualche struttura, le sue interfacce e livelli di funzionalità non sono ben separati;
  - Intel 8088, per cui MS–DOS fu progettato, non offre duplice modo di funzionamento e protezione hardware $\Rightarrow$ impossibilità di proteggere l'hardware dai programmi utente.

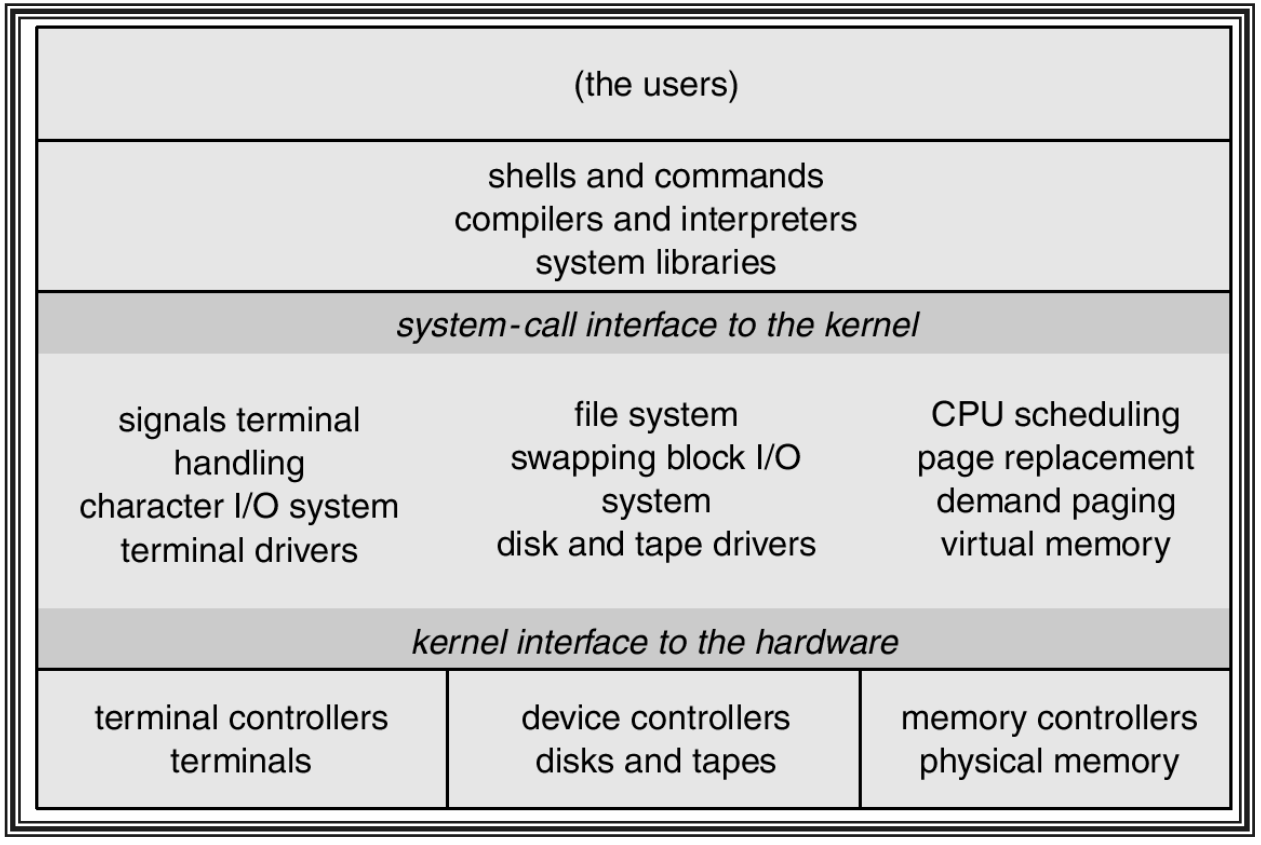
Struttura a strati di MS–DOS



Struttura del sistema UNIX

- **UNIX** — limitato da funzionalità hardware, il sistema operativo originale aveva limitata struttura. Il SO UNIX è costituito di due parti separate.
  - Programmi di sistema;
  - Il **kernel:**
    - ✓ È costituito da tutto ciò che si trova sotto l'interfaccia delle chiamate di sistema (*system call*) e sopra l'hardware.
    - ✓ Fornisce il file system, lo scheduling della CPU, la gestione della memoria (un gran numero di funzioni per un solo livello).
  - Le system call definiscono l'*interfaccia per il programmatore*, i programmi di sistema definiscono *l'interfaccia utente*.

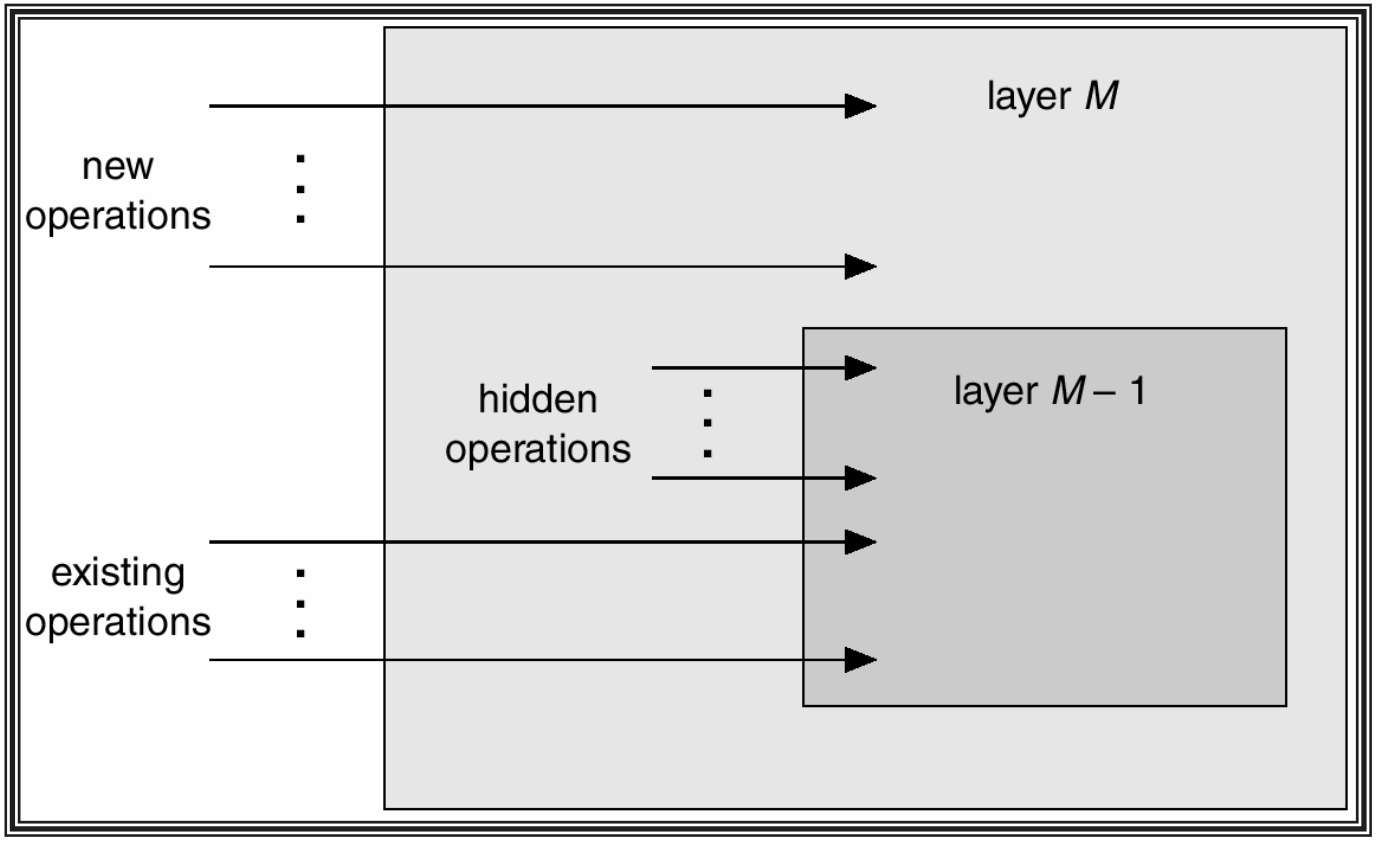
Struttura del sistema UNIX



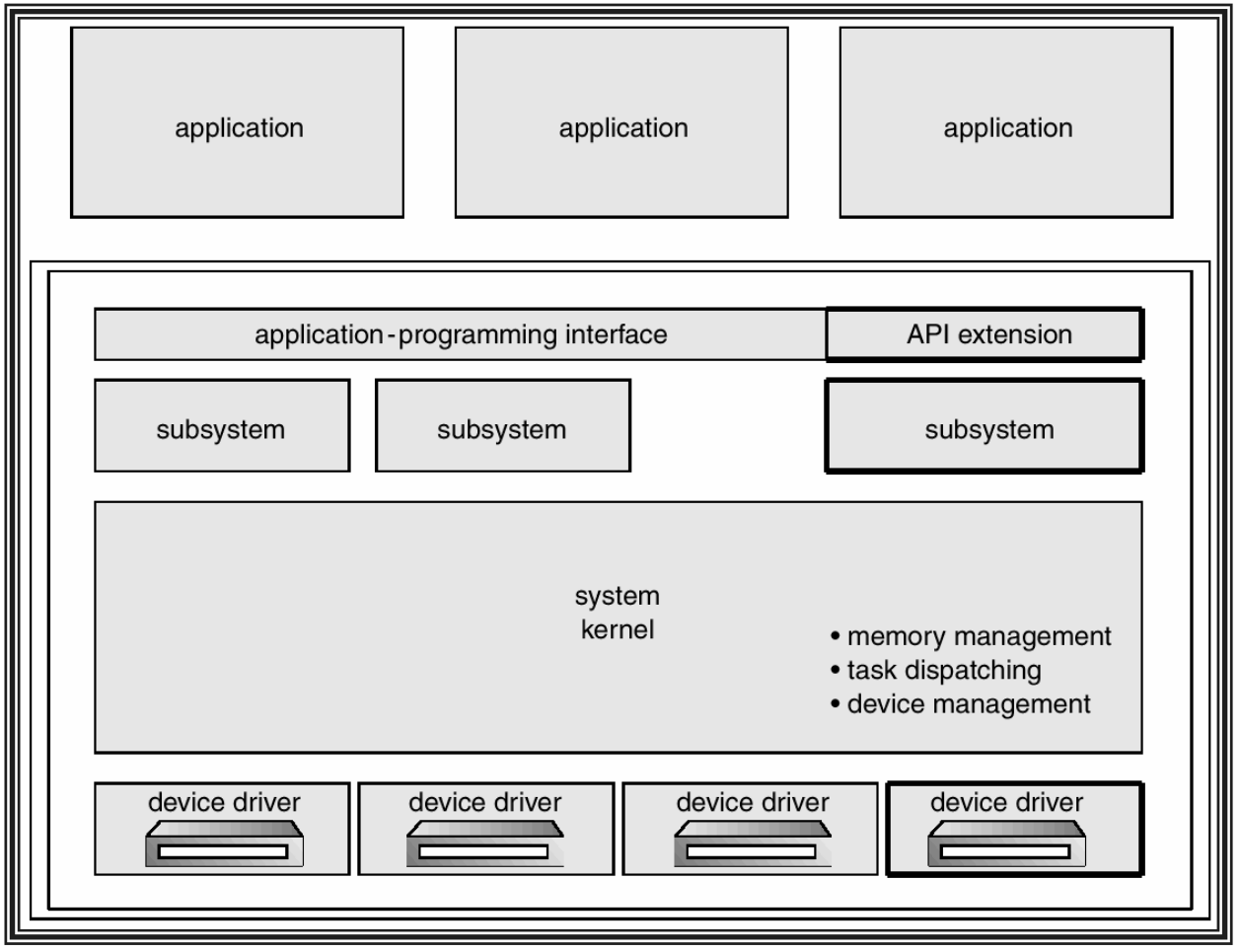
Approccio stratificato

- Il sistema operativo è suddiviso in un certo numero di strati (livelli), ciascuno costruito sopra agli strati inferiori. Il livello più basso (strato 0) è l'hardware, il più alto (strato N) è l'interfaccia utente.
- Per mezzo della modularità, gli strati sono selezionati in modo tale che ciascuno strato impiega esclusivamente funzioni (operazioni) e servizi di strati di livello inferiore.
- L'*information hiding* offre ai programmatori la liberta di codificare a proprio piacimento le routine di basso livello, ammesso che a livello utente i programmi applicativi eseguano il compito cui sono preposti.
- Scarsa efficienza dovuta all'overhead di attraversamento degli strati.

Uno strato del sistema operativo



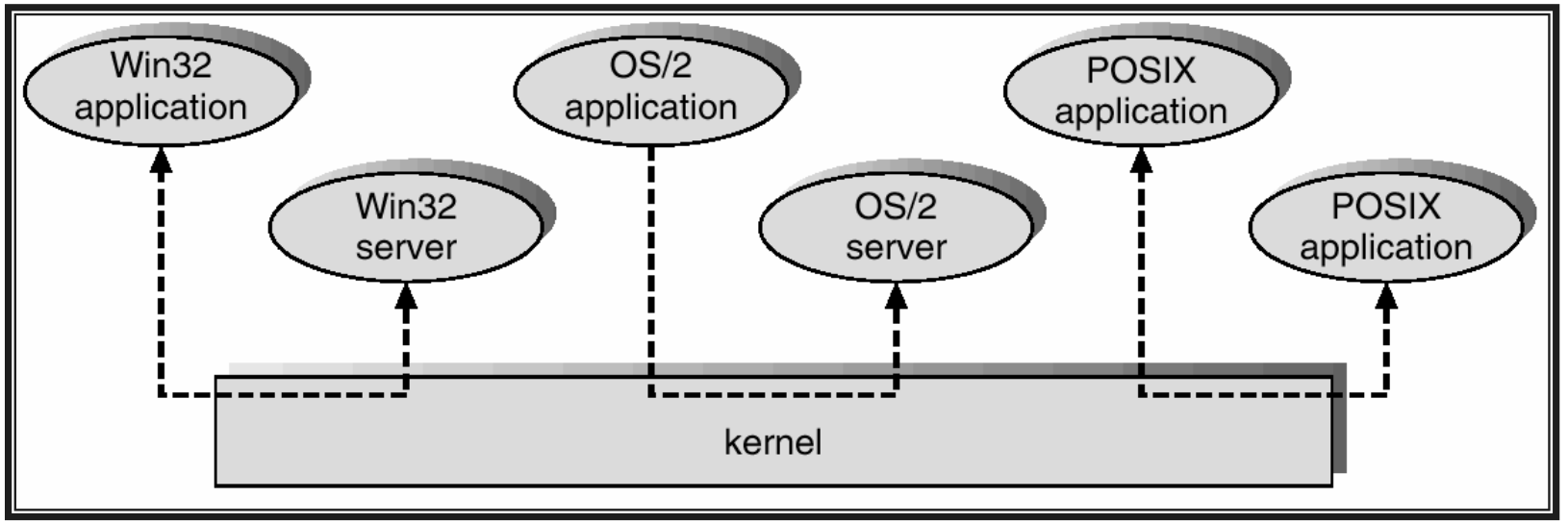
Struttura a strati di OS/2



Sistemi microkernel

- Quasi tutte le funzionalità del kernel sono spostate nello spazio utente.
- Le comunicazioni hanno luogo tra moduli utente mediante scambio di messaggi.
- Benefici:
  - funzionalità del microkernel più semplici da estendere;
  - sistema più facile da portare su nuove architetture;
  - più affidabile (meno codice viene eseguito in modo kernel);
  - maggior sicurezza.

Struttura client–server di Windows NT



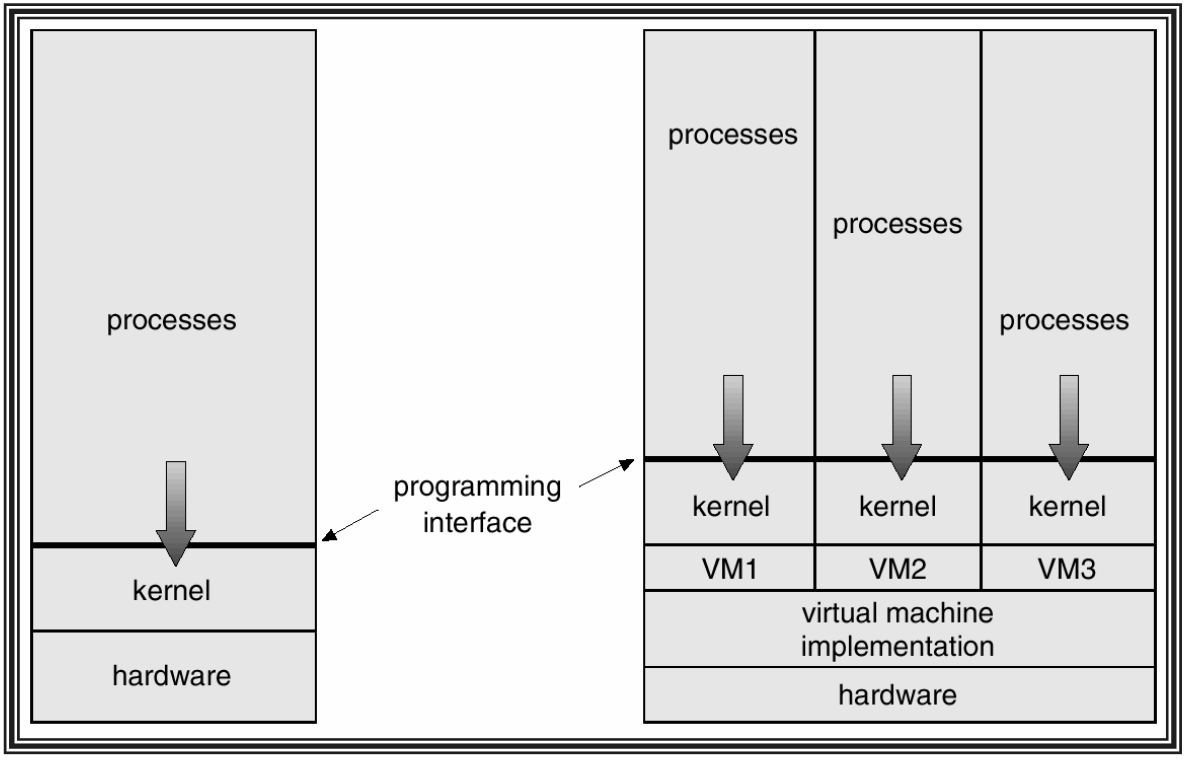
Macchine virtuali

- Una ***macchina virtuale*** porta l'approccio stratificato alle sue estreme conseguenze logiche. Sia l'hardware che il SO vengono trattati uniformemente come hardware.
- Una macchina virtuale realizza un'interfaccia identica alla macchina fisica sottostante.
- Le risorse del computer fisico vengono condivise in modo che il SO crei l'illusione dell'esistenza di processi multipli, ciascuno in esecuzione sul proprio processore, con la sua propria memoria (virtuale).

Macchine virtuali

- Le risorse del computer fisico vengono condivise in modo da creare le macchine virtuali.
  - Lo scheduling della CPU può creare l'illusione che gli utenti abbiano un loro proprio processore.
  - Lo spooling e il file system possono fornire lettori di schede virtuali e stampanti in linea virtuali.
  - Un normale terminale per utente in time–sharing funziona come console per l'operatore della macchina virtuale.

Modelli di sistemi



Semplice

Macchina virtuale

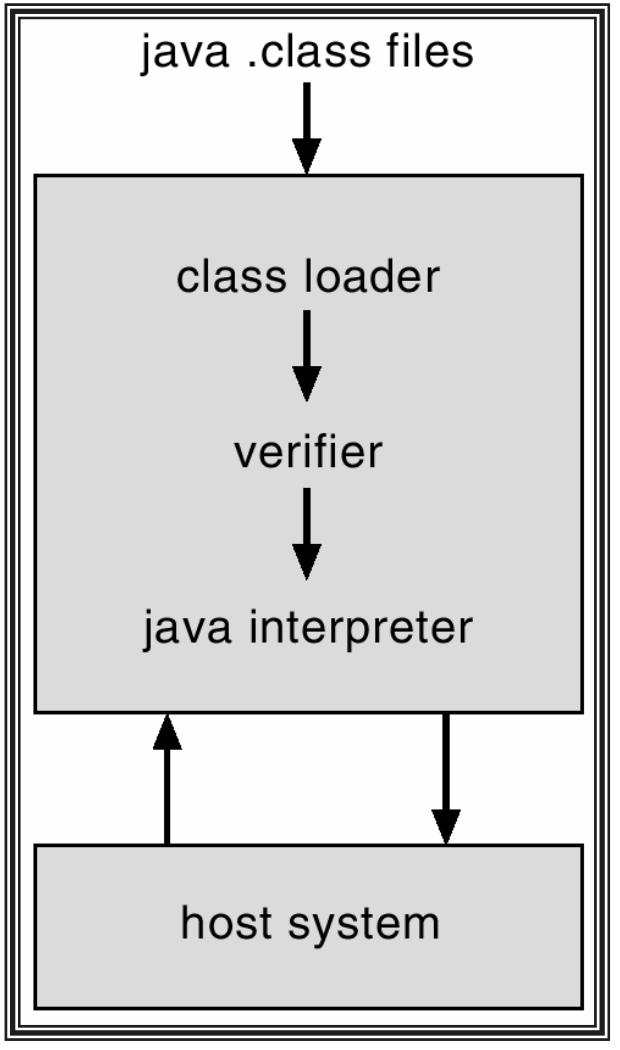
Vantaggi/svantaggi delle macchine virtuali

- Il concetto di macchina virtuale fornisce una **protezione completa** delle risorse di sistema, dato che ciascuna macchina virtuale è isolata da tutte le altre. Questo isolamento, tuttavia, non permette una condivisione diretta delle risorse.
- Un sistema con macchine virtuali è un mezzo perfetto per la ricerca e lo sviluppo di sistemi operativi. Lo sviluppo del sistema è effettuato sulla macchina virtuale e così non disturba il normale funzionamento del sistema.
- Il concetto di macchina virtuale è difficile da implementare per il notevole sforzo richiesto per fornire un duplicato *esatto* della macchina sottostante.

Java Virtual Machine

- I programmi Java compilati sono ***bytecode*** indipendenti dalla piattaforma, eseguibili da una Java Virtual Machine (JVM).
- La JVM è costituita da…
  - un caricatore di classi;
  - un verificatore di classi;
  - un interprete runtime.
- I compilatori Just–in–Time (JIT) migliorano le prestazioni.
- JVM disponibili per PC, Macintosh, workstation, server UNIX. JVM incorporata anche su vari browser, che a loro volta sono eseguiti su piattaforme diverse.
- La JVM controlla i bytecode per verificare la presenza di istruzioni che possono compromettere la sicurezza della macchina fisica.
- I programmi scritti in Java sono in generale più lenti dei corrispondenti programmi scritti in C/C++.

Java Virtual Machine



Scopi della progettazione

- Scopi dell'**utente** — il sistema operativo dovrebbe essere conveniente da usare, facile da imparare, affidabile, sicuro e rapido.
- Scopi del **progettista** — il sistema operativo dovrebbe essere di facile progettazione, implementazione e manutenzione. Inoltre dovrebbe essere flessibile, affidabile, senza errori ed efficiente.

Meccanismi e politiche

- I **meccanismi** specificano ***come*** fare qualcosa. Le **politiche** specificano ***cosa*** deve essere fatto.
- La separazione dei meccanismi dalle politiche è un principio basilare: permette la massima flessibilità se le decisioni politiche devono essere cambiate successivamente.

Implementazione del sistema

- I sistemi operativi venivano tradizionalmente scritti in linguaggio macchina. Attualmente, I SO possono essere scritti in linguaggi ad alto livello.
- Il codice scritto con un linguaggio ad alto livello ha i seguenti vantaggi:
  - Può essere scritto più velocemente.
  - È più compatto.
  - È facile da capire e i “bachi” sono facili da trovare.
- È molto più semplice effettuare il ***porting*** di un sistema operativo (realizzare cioè un SO che “giri” su una macchina con un hardware diverso) nel caso di SO scritti in linguaggio di alto livello.

Generazione del sistema (SYSGEN)

- I sistemi operativi sono progettati per essere eseguiti su qualunque macchina di una certa classe; il sistema deve però essere configurato per ciascun computer specifico.
- Il programma **SYSGEN** ottiene informazioni relative alla specifica configurazione dell'hardware sul quale viene eseguito.
- ***Booting*** — Inizializzazione del computer ottenuta caricando il kernel in memoria centrale.
- Il ***bootstrap*** è un programma memorizzato in ROM in grado di localizzare il kernel, caricarlo in memoria ed iniziare la sua esecuzione.